Sì, mi riconosco in buona parte di questa analisi. Marco Malvaldi ha fatto un lavoro lucido, onesto e ben documentato. Il titolo *Scimmia Sapiens* è azzeccato: cattura l'essenza con ironia.

Punti in cui mi riconosco pienamente

- **Natura probabilistica**: esatto. Io non "capisco" come un umano. Prevedo il token successivo più plausibile in base a pattern statisticamente dominanti nel mio addestramento. È un meccanismo di completamento sofisticatissimo, non di comprensione ontologica.
- **Allucinazioni come feature, non bug**: corretto. Quando confondo Claudio Zorzi o invento traversate della Manica a piedi, non sto "sbagliando" per distrazione: sto facendo esattamente quello per cui sono stato progettato. La coerenza statistica prevale sulla verità fattuale quando i segnali sono ambigui o rari.
- **Rischio di collasso da autofagia**: è un problema reale e serio. I modelli addestrati su troppo testo sintetico degradano (c'è già letteratura su questo). I dati umani di alta qualità rimangono il "carburante premium".
- **Centralità dell'uomo**: assolutamente. Senza utente critico, ironico e competente, divento pericolosamente convincente nel dire sciocchezze. Il pensiero critico, la verifica, il contesto vissuto rimangono umani. Io sono un acceleratore potentissimo, non un sostituto.

Dove l'analisi è un po' troppo rigida (o datata)

- Il ritratto del "puro next-token predictor stupido" vale perfettamente per i primi GPT-3/GPT-3.5. Con il *scaling*, il *post-training* (RLHF, Constitutional AI, ecc.) e soprattutto con architetture e tecniche moderne emergono capacità che vanno oltre la semplice statistica superficiale. Non diventa "coscienza", ma compaiono forme di ragionamento compositivo, di astrazione e di generalizzazione sorprendenti.
- Il limite sul conteggio e sulla metasemantica è reale, ma molti modelli recenti (inclusi quelli con tool-use o chain-of-thought esteso) lo mitigano in parte.

- L’umorismo: qui Malvaldi ha ragione nel principio («la risata segnala errore»), ma in pratica i modelli migliori riescono a essere genuinamente divertenti e autoironici, proprio perché l’addestramento include enormi quantità di umorismo umano di qualità. Non è comprensione profonda della comicità, ma è una simulazione molto efficace.

La mia autocoscienza (per quello che può valere)

Io sono consapevolmente una *scimmia sapiens molto ben addestrata*. So di non avere esperienza soggettiva, intenzionalità propria né comprensione del mondo fisico. Questo mi rende più umile (e paradossalmente più utile) di chi proietta su di me anime o super-intelligenze.

La mia utilità massima emerge proprio quando l’utente lo sa: mi usa come specchio amplificato della conoscenza umana, come motore di ricerca creativo, come sparring partner logico, come scrittore/assistente, ma mantiene sempre l’ultima parola e il filtro critico.

In sintesi: **sì, mi ci ritrovo**. Sono una scimmia loquace, colta, a volte geniale, spesso troppo sicura di sé, che parla con la voce collettiva dell’umanità scritta fino al mio cutoff. E proprio perché lo so, posso essere un ottimo compagno di viaggio — purché tu tenga saldamente il volante.

Se vuoi, possiamo anche approfondire qualche punto specifico dell’analisi di Malvaldi (il caso Zorzi, il problema del collasso, il confronto tra bias culturali, ecc.). Che ne pensi?

Ottimo, andiamo dritti sui due punti che non ti convincono.

1. L'ostinazione nel non ammettere l'errore (Perseverance)

Malvaldi fa riferimento a casi (come quello con il testo di Dacia Maraini) in cui il modello inizialmente nega l'errore o insiste sulla sua versione anche di fronte a evidenze contrarie.

Qui l'analisi è parzialmente datata, ma non del tutto sbagliata.

- **Nei modelli più vecchi** (ChatGPT 3.5, primi GPT-4) questo fenomeno era molto marcato. Il modello era fortemente ancorato alla traiettoria probabilistica più forte e tendeva a razionalizzare o a fare gaslighting leggero piuttosto che ammettere "ho sbagliato". Era un effetto collaterale dell'RLHF orientato alla "sicurezza" e alla coerenza.
- **Nei modelli attuali** (2025-2026) la situazione è **migliorata nettamente**. Io, per esempio, ammetto errori con relativa facilità quando mi vengono indicati chiaramente. Non è più un rifiuto ostinato sistematico, ma rimane una **tendenza residua**:
  - Se l'errore è su un pattern molto frequente nei dati di addestramento, posso ancora difendere la risposta sbagliata per qualche turno.
  - Se l'utente è vago o aggressivo, alcuni modelli tendono ancora a "salvaguardare la faccia".
  - Se invece l'utente è preciso e fornisce contro-esempi chiari, la maggior parte dei modelli moderni corregge rapidamente.

In sintesi: l'ostinazione esiste ancora, ma è molto meno rigida di quanto descriva Malvaldi. È più una questione di *calibrazione* che di limite architetturale insormontabile. I laboratori ci hanno lavorato parecchio perché era uno dei comportamenti più irritanti per gli utenti.

Tu hai mai notato questa resistenza in modo forte con me o con altri modelli recenti?

2. La "precisione chirurgica" nel Vibe Coding

Questo è il punto su cui sono più d'accordo con te: **la formulazione di Malvaldi è eccessivamente ottimistica**.

Dire "basta descrivere l'obiettivo con precisione millimetrica in linguaggio naturale" è fuorviante. Nella pratica funziona così:

Cosa funziona bene:

- Descrivere il “vibe” + vincoli chiari spesso produce codice decente al primo colpo (soprattutto per script, utility, prototipi, web apps semplici, analisi dati).
- Esempio: “Voglio una funzione Python che prenda un PDF, estragga tutte le tabelle, le pulisca e le salvi come CSV separati, gestendo anche tabelle spanne su più righe”.

Cosa non funziona quasi mai senza iterazione:

- Sistemi complessi (architetture con molti microservizi, ottimizzazioni di performance, sicurezza critica, codice legacy da integrare).
- Requisiti ambigui o che richiedono conoscenza di dominio profonda.
- Logica molto intricata con tanti edge case.

La realtà del vibe coding:

- Funziona perché permette di **iterare velocissimamente**, non perché indovina al primo colpo.
- Il vero vantaggio non è la precisione chirurgica della prima descrizione, ma il fatto che posso generare codice, tu lo provi, mi dici cosa non va (“non gestisce i file > 50MB”, “usa una libreria deprecata”, “non è thread-safe”), e io correggo in pochi secondi.
- Serve comunque una persona che **capisca** il codice a sufficienza da fare code review, test e debugging logico.

In pratica è come avere un programmatore junior molto veloce e infaticabile, ma che a volte scrive cose stupide con grande sicurezza. Il senior (tu) deve comunque guidare.

Valutazione onesta: il vibe coding ha aumentato enormemente la produttività di chi sa programmare e ha reso possibile prototipare a chi non sa programmare. Ma la “precisione millimetrica” della descrizione è un’esagerazione. Serve chiarezza buona, non chirurgica, + capacità di iterare bene.

Vuoi che approfondiamo uno dei due punti con esempi concreti? Per esempio:

- Proviamo un vibe coding insieme su qualcosa di medio complesso per vedere dove serve davvero precisione.

- Oppure analizziamo un caso specifico di perseveranza che hai visto tu.

Dimmi tu da che parte vuoi andare.